CHAPTER

7

THE ROLE OF THE SYSTEM ADMINISTRATOR

After reading this chapter and completing the exercises, you will be able to:

- ♦ Explain the work of a system administrator
- ♦ Discuss the responsibilities of a system administrator
- ♦ Use basic system administration commands in Linux

In the previous chapter you learned about the Linux command-line environment (the shell) and how to use different text editors and text-processing tools to create scripts and work effectively at the command line.

In this chapter you will look at the role of a Linux system administrator. The topics in this chapter lay the foundation for the rest of this book, which covers in detail the separate tasks performed by a system administrator.

WORKING AS A SYSTEM ADMINISTRATOR

The job of a system administrator is to make technology work and continue to work. While others may develop programs or devices with great potential, the system administrator ties them into complete, operational systems that can increase productivity, lower costs, or otherwise benefit those who use the technology. The system administrator keeps these systems running efficiently as new pieces are added, changes occur, and reconfigurations and failures alter the face of the original systems. The job of the system administrator is primarily practical. It requires perseverance, patience, curiosity, creativity, and technical knowledge. To be truly successful as a system administrator, you must continue to increase both the breadth (number of subjects) and depth (expertise in a subject) of your technical knowledge. If you don't, new problems will come along that you won't know how to solve. At the same time, you will lack the ability to integrate new technologies into your systems or to determine how they apply to your environment.

A system administrator generally works as part of the **Information Systems (IS)** or **Information Technology (IT)** Department of an organization. In a large organization, this group reports to a **chief information officer (CIO)**. In smaller organizations, a group of system administrators might consult other company officers to make decisions about information technology. The IS or IT Department is concerned only with internal information systems. In technology organizations (such as companies that develop software or sell computers or telecommunications equipment), the team that develops software and hardware for sale to others is not generally part of the IS or IT Department. Figure 7-1 shows the position of a system administrator in a typical small or large company.

In addition to working with technology, system administrators also work with people. They may work primarily with a group of technical colleagues in their department, but they also are likely to interact with most of the organization as they answer questions, solve technical problems, train users, install software, and so forth. In larger organizations, the tasks of working with end users and maintaining the systems are divided into different areas. For example, the IS team may manage the servers, while the **Help Desk** team directly solves problems for **end users** (those who use computer systems to accomplish their daily work). In such an environment, you, as a system administrator, can focus on the particular area that best suits your interests. The same technical knowledge and problem-solving ability are required for both types of work.

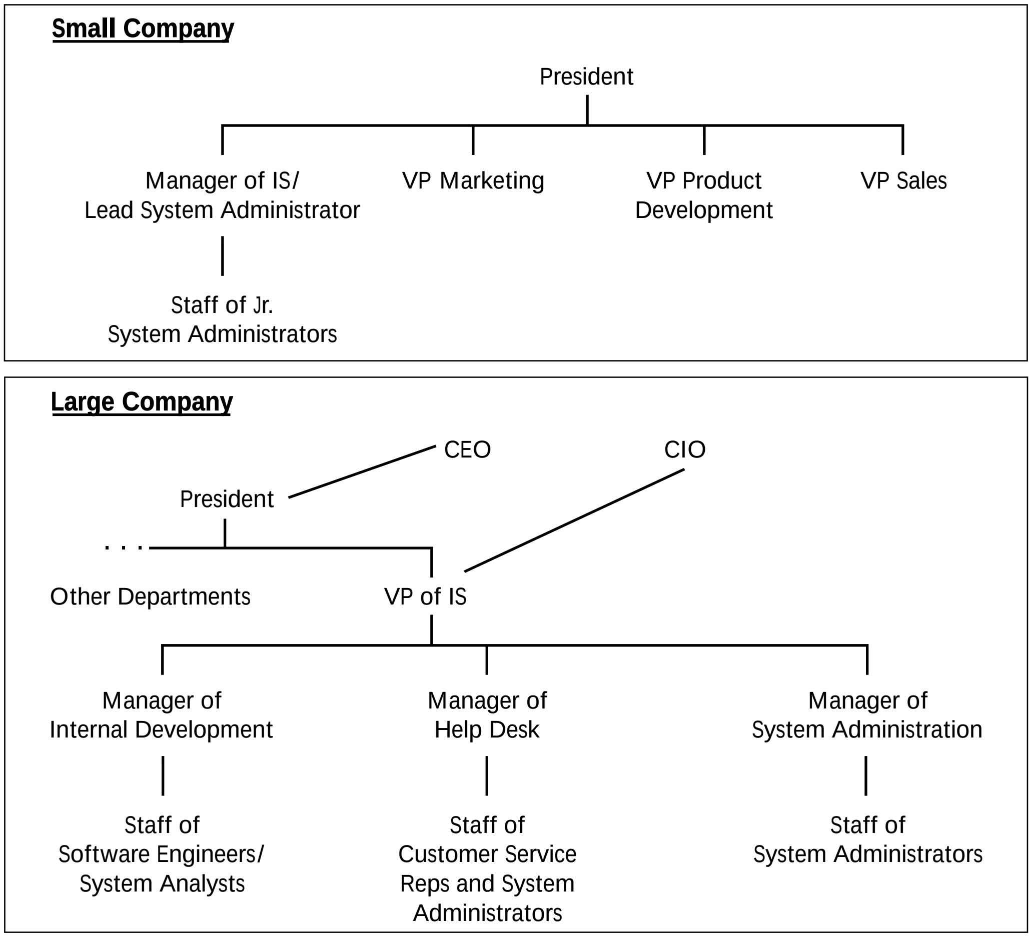

Small Company
President
Manager of IS/ Lead System Administrator
VP Marketing
VP Product Development
VP Sales
Staff of Jr. System Administrators
Large Company
CEO
CIO
President
. . .
Other Departments
VP of IS
Manager of Internal Development
Manager of Help Desk
Manager of System Administration
Staff of Software Engineers/ System Analysts
Staff of Customer Service Reps and System Administrators
Staff of System Administrators

Figure 7-1 Role of the system administrator within large and small companies

Tasks of a System Administrator

As a system administrator, the tasks for which you are responsible can vary considerably based on factors such as:

- Your expertise and specific job position
- Your seniority in an organization
- The size of the organization

The following list describes in detail some of the common tasks a system administrator performs.

- Creating new user accounts and making changes in user accounts, such as granting new access permissions as assignments change.
- Maintaining system hardware, including installing new hardware as part of increasing system capacity, replacing damaged systems, or upgrading obsolete components.
- Training end users to effectively use new systems, software, or procedures.
- Performing other occasional or recurring tasks that keep the system running smoothly. Some of these are routine, such as backing up files; others require more creativity, such as determining why system response time has slowed, or tracking down an intruder from the Internet.
- Documenting the system so that other system administrators can understand your work. This might include informing others of how applications are configured, where back-up files are stored, and which users have had specific problems with certain hardware or applications. This task is often related to the next one.
- Defining procedures and policies related to how systems are administered at your site. Among other things, a system administrator might need to define back-up procedures, privacy and security policies, user guidelines, or a disaster plan. (All of these topics are discussed in future chapters.)
- Recovering from emergencies to get a system running again after a power outage, hardware failure, employee problem, or natural disaster.
- Planning a system. When working within small organizations or departments, or as you gain experience in a large organization, you may be asked to decide on new hardware purchases or plan for future system components or designs to meet anticipated needs.

In addition to the core tasks listed above, you may be asked to:

- Inform management of potential technical needs for upgrades.
- Watch for security threats and implement remedies when possible.
- Keep yourself up-to-date regarding new developments in your field. Staying current allows you to take advantage of new developments that could benefit the employer's information technology strategy and increase your value to your employer (and to future employers).

Ethics and the System Administrator

Working as a system administrator involves many ethical issues that may not be evident at first. As a system administrator you have control—full or partial—over an organization's computer systems. Implicit in this responsibility is a great deal of trust on the part of both the company (its officers, managers, and owners) and the individual employees who use the systems that you manage. The way you view this trust will likely determine how effective you are as a system administrator.

Employers pay you to maintain their systems in a way that contributes to the success of their organization. Your role also has an important effect on individual users. Although you may be working behind the scenes most of the time (and probably should be if things are running smoothly), remember that your fellow employees count on your work in order to do theirs. A lack of preparation or accuracy on your part can lead to companywide downtime, corrupted or lost files, malfunctioning printers, and so forth. As a result, none of the other employees can be productive. Everyone in a modern office relies on the work of a good system administrator every workday.

Along with this control over the working lives of others comes the potential for abuse of this power. For example, as the system administrator with root privileges on the Linux server, you have the power to:

- Read people's e-mail and the files in their home directories
- Alter company or personal files
- Send falsified messages as if they came from other users
- Erase ("lose") any files on the system
- Delay fixing a system problem or helping an employee with a simple question
- Neglect security measures that would protect sensitive data

These actions are unethical because they invade others' privacy and impede the work of your employer. Many are also illegal and would make an unscrupulous system administrator subject to prosecution. But as you may realize, many unethical actions are likely to go undetected, especially if you are the sole person in a company with expertise in Linux. You should decide at the outset on a few rules that can guide you in your relationships with employers and fellow employees whose systems you manage. Your rules might include statements like these:

- I realize that I know more about the systems I manage than others, but I also realize that they know more about their job functions and what they need from their computer systems.
- I will never read files that do not belong to me personally unless required to do so as part of a legal order or to comply with a publicly acknowledged company policy.
- I treat other employees as my clients. Success as a system administrator depends on their satisfaction regarding how I meet their needs for information technology.

Occasionally, a system administrator may decide to configure systems in such a way that no one else can figure out how the system is configured or used. This is sometimes done in the name of job security: "They can't fire me," this kind of system administrator reasons, "or the entire company will have to shut down."

In fact, however, your best route to success as a system administrator (not to mention peace of mind) will come through making your employer successful. This allows you to grow professionally, with additional responsibility and technical opportunities. If you train yourself well, you need never feel compelled to make implied threats of holding your employer "hostage" because you are the only person who can maintain the computer systems. Remember these two rules:

- Good jobs are always available for well-trained technical people; hence job security should not depend on work at a single company. Build a reputation as both a technical expert and a personable employee to make future employers eager to hire you and past employers sorry they lost you.
- If you haven't trained yourself well, you're not worth keeping as an employee. Your employer can then replace you with someone who is not being territorial under the guise of "job security." The true expert will always be able to set up efficient, standardized, well-documented systems and have a solid career based on managing those systems.

To read more about working as a system administrator, you should also visit the System Administrators Guild (**SAGE**) at *http://www.usenix.org/sage/*. SAGE is part of the USENIX group, an organization for people who work with advanced computing systems that provides tremendous resources to system administrators. The SAGE Web site contains information about:

- Jobs and salary profiles
- Local user groups
- Technical information
- Events where system administrators gather for technical conferences

PRINCIPLES OF MAINTAINING A LINUX SYSTEM

Compared to the other types of technical work demanded of a system administrator, learning about Linux is especially rewarding. Whereas some technical topics relate to mastering a specific graphical tool or proprietary method, knowledge that you gain about Linux is generally applicable to a wide range of systems and situations. Although learning Linux well can be a challenge, that knowledge carries over to other systems. For example, if you learn about the Domain Name Service (DNS) on Linux, you will find that the knowledge applies to DNS servers on practically any system in the world. Knowledge of Linux also forms a strong foundation for learning about related topics such as TCP/IP routing or NIS+. Or suppose you learn about configuring an Apache Web server on Linux. Other Web servers may have graphical interfaces that are easier to configure, but the concepts relating to how a Web server operates and the options you learn about in detail as you work in Linux will apply to nearly every Web server available.

As you become familiar with the Linux tools used for system administration, you may notice that they are different in fundamental ways from tools used on non-UNIX-like operating systems such as Windows NT. The history of UNIX (and thus of Linux development by association), followed a very different path from Windows NT. The result is that certain methods of solving problems have been developed on Linux and UNIX systems. As you learn about these methods, you will be better able to use the tools that Linux provides to keep your Linux systems running efficiently with the least amount of work and the fewest headaches.

Many of the principles outlined in this chapter have been developed over the 30-year history of UNIX and Linux technology. Thirty years ago, computers were much slower, more expensive, and more difficult to use (no graphical interfaces were available until fairly recently). UNIX (and Linux) were originally designed for these systems. As a result, Linux is generally more efficient in using limited system resources.

When designing an operating system for these early computers, UNIX and Linux developers were forced to create extremely efficient operating systems. For example, special files known as shared libraries allowed multiple programs to use the same set of functions stored in memory. The goal of efficiency in Linux is reflected in the way the various system administration tasks are organized. Some key facts to notice about Linux system administration are the following:

- Plain-text files configure the system. Individual files control each program or service.
- Everything on the Linux system is accessed as if it were a file, including devices and remote computer systems.
- The entire system is designed to be used by multiple users.
- Linux command-line utilities are usually small and simple in function, being designed to do just one task very well. They have the capacity, however, to be connected with other commands to complete more complex tasks.

The sections that follow describe some of these traits in more detail.

Linux Configuration Files

A full-featured Linux system may support hundreds of users and include thousands of programs on its hard disks. Many of these programs are system or network services, such as a Web server, an e-mail server, or a Samba server (to allow SMB clients like Windows 98 to connect to Linux). Other programs might include a graphical **utility** (a program used for system administration) on a Linux desktop such as Gnome or KDE, or an end-user program such as WordPerfect for Linux or ApplixWare for Linux.

Each of these programs and system services creates and maintains its own set of configuration information. The configuration files for programs run by the system administrator (and available to all users), such as a Web server, are normally stored in the `/etc` directory. Configuration files for programs that are used by only a single user are stored in that user's home directory. Programs that are used by all users on a system often include default configuration information that is applied when any user runs the program, plus user-specific options that are stored in a user's home directory.

Configuration files for Linux programs and services do not follow a well-defined pattern or naming convention. Some configuration files end with the word `conf`, for *configuration*. Others end with `rc`, for *run control*. The names of some configuration files show no indication of what the file is used for—as the system administrator, you must simply know which file to look for. Some of the configuration files used by Linux are listed in Table 7-1. The exact location of these files varies slightly depending on the version of Linux that you use—most are located in the `/etc` directory or one of its subdirectories.

Table 7-1 Configuration Files Used by Linux

Filename	What the file configures
`XF86Config`	XFree86 graphical system
`smb.conf`	Samba server
`httpd.conf`	Web server
`resolv.conf`	DNS name resolver (selects a Name server to access)
`hosts`	Hostnames and IP addresses used for networking access
`xinitrc`	Programs that start along with the graphical system
`ftpaccess`	FTP server
`lilo.conf`	LILO boot loader
`bashrc`	Configuration script that runs when starting a shell
`passwd`	User account names and configuration information

TIP

You can immediately begin learning about the content and format of any of the configuration files in Table 7-1 by entering the command `man 5` *`filename`*.

Advantages of Multiple Plain-Text Configuration Files

The historical forces that have made Linux so powerful have also resulted in numerous and diverse configuration methods: software developers are not required by any authority to follow a specific pattern. In fact, UNIX systems have always used text-based configuration files. Using **plain-text configuration files** has the following advantages:

- You can easily write a program to manipulate the configuration of a program or service, because this involves basic text string manipulation.

- Each configuration file is small and independent, which can create a more efficient use of resources to update or query the configuration of a program.
- If one configuration file becomes corrupted, other configuration information is not affected—Linux configuration has no single point of failure or vulnerability.
- Developers creating programs can create new configuration designs to meet their needs, without being constrained to fit an existing configuration architecture.
- You can use a single tool (any text editor) to configure the most complex features of any program or system service.

Disadvantages of Multiple Plain-Text Configuration Files

Using text-based configuration files also has the following disadvantages:

- The system administrator must learn multiple configuration formats, some of which are highly complex, in order to set up and maintain a Linux system.
- New programs cannot take advantage of an existing configuration method or architecture to speed development.
- Text-based configuration files can be organized according to extremely complex rules, which often allow for many possible methods of expressing a configuration. These rules vary for each configuration file (and for possible new programs coming out all the time). These factors make it very difficult to create graphical configuration tools to make Linux configuration easier to learn or manage.

Despite the difficulty involved, many serious efforts have been made to create graphical configuration tools to ease the task of learning to configure Linux, much as many UNIX systems have an overall system administration interface. Some of these development efforts are aimed at configuring a single program, such as the Apache Web server. The Comanche project is one example of this. Figure 7-2 shows Comanche, a graphical utility for configuring many parts of the Apache Web server.

The Samba suite is another example of a service that you can configure using a graphical configuration tool for configuring Linux. Many graphical tools are available to help you set up Samba, including Ksamba and GTKSamba, which sets up a Samba file and print server. GTKSamba is illustrated in Figure 7-3.

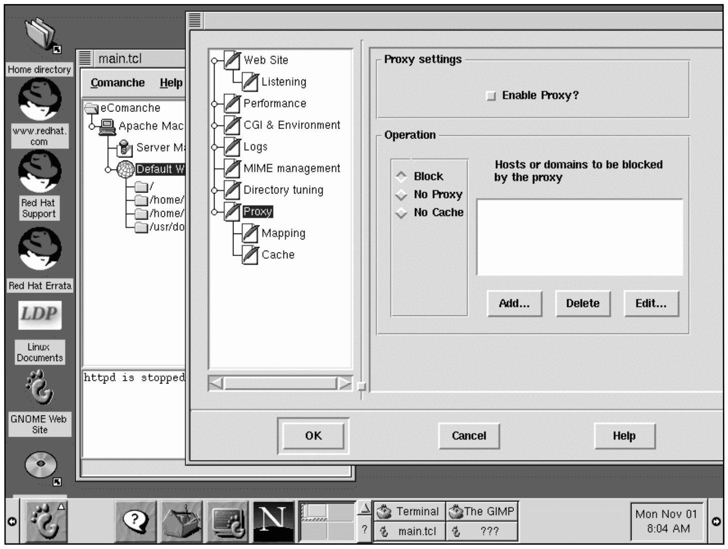
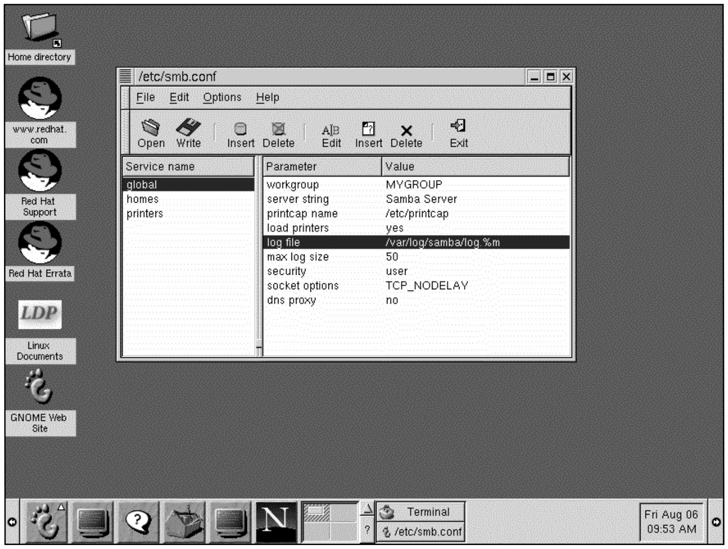


Figure 7-2 The Comanche graphical utility



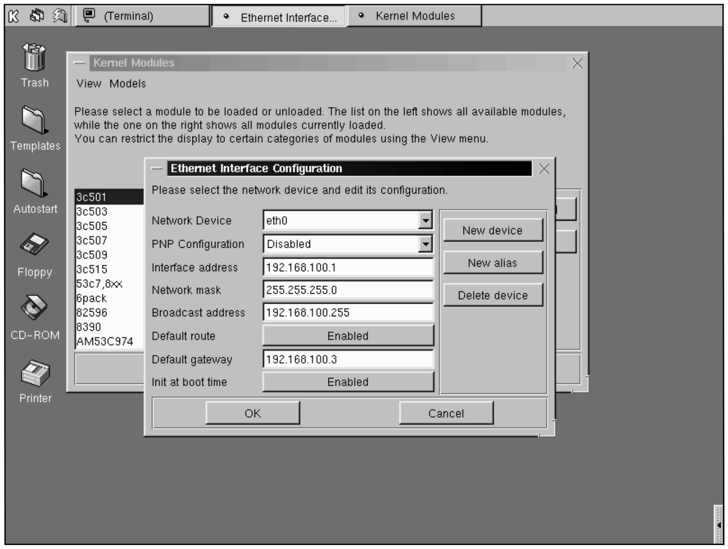
Service name
global
homes
printers

Parameter	Value
workgroup	MYGROUP
server string	Samba Server
printcap name	/etc/printcap
load printers	yes
log file	/var/log/samba/log.%m
max log size	50
security	user
socket options	TCP_NODELAY
dns proxy	no

Figure 7-3 GTKSamba

Other tools have been designed to provide a framework for all Linux configuration and maintenance. At least three of these programs have been sponsored by Linux vendors in an effort to make Linux easier to use. These three are:

- **COAS**, the Caldera Open Administration System, sponsored by Caldera Systems (shown in Figure 7-4)



Please select a module to be loaded or unloaded. The list on the left shows all available modules, while the one on the right shows all modules currently loaded.
You can restrict the display to certain categories of modules using the View menu.

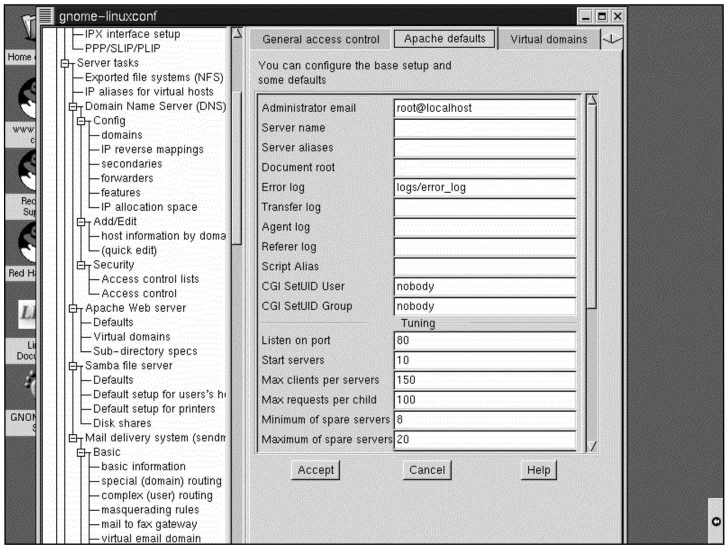
3c501
3c503
3c505
3c507
3c509
3c515
53c7,8xx
6pack
82596
8390
AM53C974

Please select the network device and edit its configuration.

Network Device	eth0
PNP Configuration	Disabled
Interface address	192.168.100.1
Network mask	255.255.255.0
Broadcast address	192.168.100.255
Default route	Enabled
Default gateway	192.168.100.3
Init at boot time	Enabled

Figure 7-4 COAS, a graphical administration tool from Caldera Systems

- **LinuxConf**, sponsored by Red Hat Software (shown in Figure 7-5)

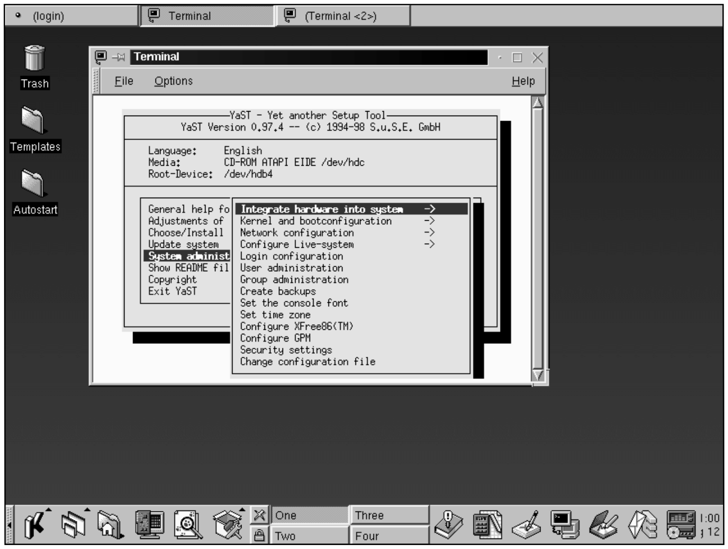


You can configure the base setup and some defaults

Administrator email	root@localhost
Server name	
Server aliases	
Document root	
Error log	logs/error_log
Transfer log	
Agent log	
Referer log	
Script Alias	
CGI SetUID User	nobody
CGI SetUID Group	nobody
Tuning	
Listen on port	80
Start servers	10
Max clients per servers	150
Max requests per child	100
Minimum of spare servers	8
Maximum of spare servers	20

Figure 7-5 LinuxConf, a graphical administration tool from Red Hat Software

- **YAST**, sponsored by SuSE (shown in Figure 7-6)



```
——————————YaST - Yet another Setup Tool——————————
     YaST Version 0.97.4 -- (c) 1994-98 S.u.S.E. GmbH

   Language:     English
   Media:        CD-ROM ATAPI EIDE /dev/hdc
   Root-Device:  /dev/hdb4

   General help fo  Integrate hardware into system    ->
   Adjustments of   Kernel and bootconfiguration      ->
   Choose/Install   Network configuration             ->
   Update system    Configure Live-system             ->
   System administ  Login configuration
   Show README fil  User administration
   Copyright        Group administration
   Exit YaST        Create backups
                    Set the console font
                    Set time zone
                    Configure XFree86(TM)
                    Configure GPM
                    Security settings
                    Change configuration file
```

Figure 7-6 YAST, a graphical administration tool from SuSE Linux

Success in these efforts to create graphical configuration tools has been hampered because Linux configuration options are too varied and complex to allow a simplistic tool to be of much use in real life. Complete tools require substantial development time; but because Linux programs don't remain static for long, any developer who did invest the necessary time and money to create a truly comprehensive administration tool would find that the tool was out of date upon its release. In addition, many Linux administrators prefer to work directly in text files, feeling that they have more control over the configuration without the system overhead of a graphical interface.

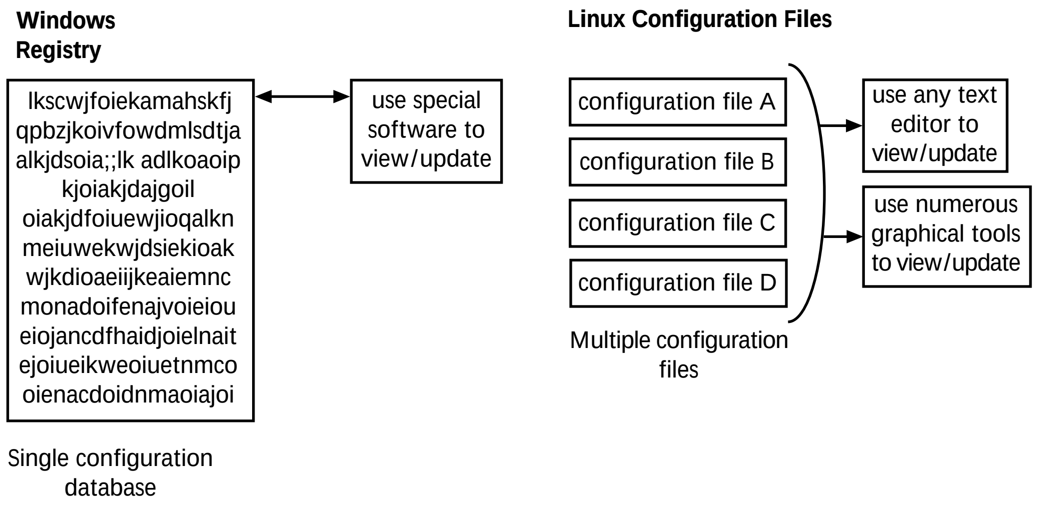
This introduces the question: Why not use graphical utilities to configure Linux where possible, especially while learning Linux, since more advanced features won't be immediately needed?

The answer is perhaps more philosophical than practical at this stage: because someone has to know how the system *really* works. As the system administrator for a Linux-based server or network, you will often be called on to solve problems that cannot be solved by even advanced graphical tools. These problems may cross boundaries between different programs, and they may involve networking activity that you have no control over, or require making minor adjustments to configuration files that would not be available in a graphical tool. If you can't get "under the hood" of the system, as the saying goes, and adjust all possible program parameters, your ability to keep a system running smoothly is much reduced. You are left instead clicking buttons, wondering what is really happening on a system that continues to have a problem you cannot diagnose or repair.

TIP

Linux and most Linux programs provide the added benefit of allowing you to review the program source code. Using this method of last resort, you can fix anything, given enough time. But before you can take advantage of this option, you must start by learning *how* things work, not simply which buttons to click in order to complete rote or simple tasks.

The numerous plain-text configuration files in Linux provide access to all features of Linux programs; once you are familiar with these files, you can use them to solve any problem. Other operating systems put information about system services and resources into a single configuration file. An example of such a file is the Windows Registry. Figure 7-7 shows how the Windows Registry differs from the multiple configuration files used by Linux.



Windows Registry
lkscwjfoiekamahskfj qpbzjkoivfowdmlsdtja alkjdsoia;;lk adlkoaoip kjoiakjdajgoil oiakjdfoiuewjioqalkn meiuwekwjdsiekioak wjkdioaeiijkeaiemnc monadoifenajvoieiou eiojancdfhaidjoielnait ejoiueikweoiuetnmco oienacdoidnmaoiajoi
use special software to view/update
Single configuration database
Linux Configuration Files
configuration file A
configuration file B
configuration file C
configuration file D
use any text editor to view/update
use numerous graphical tools to view/update
Multiple configuration files

Figure 7-7 Registry vs. multiple configuration files

Understanding Files, Devices, and Processes

Managing a Linux system includes managing all of the resources of the system: the file system, the devices, and the processes. As you learned in Chapter 4, the default Linux file system is large and complex, with predefined locations for most types of files. The task of locating, creating, moving, and deleting files is part of every system administrator's day. The later section "Using Basic System Administration Tools" describes many of the utilities you can use to manage the Linux file system. The following sections define the terms *devices*, *processes*, *programs*, and *thread*.

Devices

Devices are an important part of your system maintenance responsibilities. All of the resources at your disposal in Linux—the hard disk, modem, mouse, system memory—must be accessed using the appropriate Linux method. Linux accesses devices by assigning them a filename in the `/dev` subdirectory, as described in Chapter 4. Later chapters, such as Chapter 14, describe many tools that interact with the physical devices that are part of your computer.

Some of these devices are accessed directly. For example, when you configure a printer as described later in this book, you may refer to the `/dev/lp0` device name. This pathname indicates your computer's first parallel port. Other devices are accessed indirectly. For example, to use a hard disk, you configure access to the file system on that hard disk by referring to the device, such as `/dev/hda1`, and linking it to a standard directory path, such as `/usr` or `/home`. Linux users do not access the disk via the `/dev/hda1` device name, but by referring to the standard directory path (the `/usr` or `/home` directory in this example) to which access has been configured.

Processes

Processes are the individual programs running on a Linux system. Because Linux is a multitasking operating system, many programs can be running on a Linux system at the same time. Chapter 10 describes in detail how you can manage multiple processes to make the most efficient use of system resources for all users. The section "Using Basic System Administration Tools," later in this chapter, describes a few common utilities that provide information about the processes running on Linux at any moment.

Although process is a precise term used to describe a task that the Linux kernel is running, several other terms are commonly used to refer to various types of processes. To avoid confusion, review the related terms in the sections that follow.

Program

The word **program** is a vague term for a piece of software that executes on the Linux system. A program may be composed of many different processes or tasks that Linux manages in concert to accomplish an overall goal, or a program may have just one process. The terms program, utility, tool, and software package are all used interchangeably when discussing software that runs on Linux. All of these terms are imprecise compared to using the term process, but they serve the purpose of outlining what is being described.

Daemon

A **daemon** is a background process. It normally runs continually, but it does not have any visible output. An example of a daemon is an FTP server. It processes incoming requests, sending back files as needed, but it never displays anything on the Linux screen. Instead, information on the activities of a daemon is normally recorded in a log file (`/var/log/xferlog` in the case of the FTP daemon). Many daemons are usually running on a Linux system at any moment. The name of a daemon program usually ends with the letter *d*. Some of the daemons you can expect to see running on your Linux system after a default installation include those listed in Table 7-2.

Table 7-2 Daemons Running on Linux

Daemon name	Description
`crond`	Runs scripts at scheduled times (as described in Chapter 12).
`httpd`	Responds to Web browser requests using the HTTP protocol.
`inetd`	Watches for incoming requests of many types and starts the appropriate daemon to respond to the request. (Requests to Internet services such as FTP, Telnet, Finger, Talk, and Gopher are normally handled through `inetd`.)
`syslogd`	Records information from running programs to the system log file `/var/log/messages`.

Thread

A **thread** is a piece of a process (or a piece of a daemon, since a daemon is a type of process). Threads are most commonly used in multiprocessor environments (computers with more than one CPU installed). A single task normally performed by a process in sequential fashion can be split into multiple threads, or subtasks, that can be accomplished in parallel by multiple CPUs working at the same time. The distinction between processes and threads is not important for most system administration work. Instead, the term process is used in most cases that don't involve programmers developing software for multiprocessor computers.

Multiple Users, Multiple Processes

Linux was designed from its initial stages to be a multiuser operating system. As you have already seen, during the installation of Linux, you must create user accounts before any user can log on to use the system. No one can enter commands at a Linux command line without first entering a valid username and password to log on.

Each user account can execute multiple programs (start many processes). Each of these processes is associated with the user that started it and can be managed by the system administrator accordingly. For example, in Chapter 10 you will learn how to assign a higher priority to all of a user's processes so that they are executed faster. You will also learn how to stop (kill) a single process that might be consuming too many system resources or that has stopped working correctly.

Because a Linux system often supports many users and each user runs many processes, the management of users and processes forms an important part of system administration.

Using Small, Efficient Utilities

Linux utilities (most of which are based on UNIX utilities that have been used for decades) usually perform only a single task. The design goal for these system utilities is to do a single task, offer flexibility in how to perform the task, and do it very quickly (with the most efficient use of system resources—CPU time and disk space).

To provide flexibility, Linux commands often have numerous options that you can add to modify the basic operation of the command. For example, the `ls` command used to list files (described in the next section) supports over 40 options. You can select these options by including them after the command name.

Almost all Linux commands use the same format for including options. Each option is represented by either a hyphen followed by a single letter or two hyphens followed by a word describing the option. If single letters are used to select options, they can be combined after a single hyphen. If full-word descriptions are used to select options, each must be written out separately. In both cases, the options are listed before any filenames or other parameters to the command.

TIP Some Linux utilities, such as `ps` and `tar`, described later in this chapter, use single-letter options without a hyphen preceding them.

Table 7-3 lists 10 common `ls` command options.

Table 7-3 Common Options of the `ls` Command

Single-letter format	Full-word format	Description
`-a`	`--all`	Lists all files in a directory, including hidden files (files that start with ".")
`-l` Note: Use a lowercase letter *L* for this option.	`--format=long`	Prints not only the names of items in a directory, but also their sizes, owners, dates of creation, and so forth
`-C`	`--format=vertical`	Displays items in sorted columns
`-r`	`--reverse`	Reverses the sorting order of the items being listed
`-t`	`--sort=time`	Sorts items being listed by their timestamp rather than alphabetically
`-S`	`--sort=size`	Sorts items being listed by their size rather than alphabetically
none	`--color`	Displays files color coded according to type
none	`--help`	Displays help text with an abbreviated options list
`-I` Note: Use an uppercase letter *i* for this option.	`--ignore` *`pattern`*	Does not display items matching the pattern given
`-R`	`--recursive`	Lists the contents of all subdirectories as well as the current directory, showing the entire directory tree
`-i`	`--inode`	Prints the index number for each file to the left of the filename

Caution

Both the names of Linux commands and their options are case sensitive. The `-r` option and the `-R` option are both valid and have very different meanings.

You can combine options in several ways, as the examples in Table 7-4 show.

Table 7-4 Combining Command Options

Command example	Description of results
`ls -laSr`	Lists the contents of the current directory, including all files (`-a`), in long format (`-l`), sorted by size (`-S`), in reverse order (`-r`).
`ls -l -a -S -r`	Same as the previous example.
`ls -R --color`	Lists the contents of all subdirectories (`-R`), color coding each item shown (`--color`). In this example, no single character option for `--color` is supported, so the two options cannot be combined.
`ls --format=vertical --sort=time -ai`	Lists all files (`-a`), including their index numbers (`-i`), in a vertical column, sorting them by their creation time and date.

Standard Input and Output

Most input and output in Linux is done using standardized channels. Normally input comes from the keyboard and output goes to the screen. These channels can be redirected, however, using **redirection** operators. The redirection feature gives you great flexibility in using Linux utilities.

When a program expects input such as a line of text, it reads that information from the **standard input** channel (abbreviated STDIN). Normally, the STDIN data comes from the keyboard. But you can redirect input so that the program reads data from a file or from another program instead of the keyboard.

When a program writes output, it normally writes to the **standard output** channel (abbreviated STDOUT). This information is normally written to your console screen in the window where the program was launched. The STDOUT data can be redirected, however, so that it is written directly to a file or sent to another program.

A third standard channel, called **standard error**, is also used. Error messages are written to standard error (abbreviated as STDERR) separately from STDOUT in case STDOUT has been redirected. Of course, the output of STDERR can also be redirected to a special location such as an error log file.

A special tool related to redirecting communication between programs is called a pipe. A **pipe** connects the output channel of one command to the input channel of another command. Pipes are used to connect the output of one application to the input of another application. Figure 7-8 shows how this works conceptually.

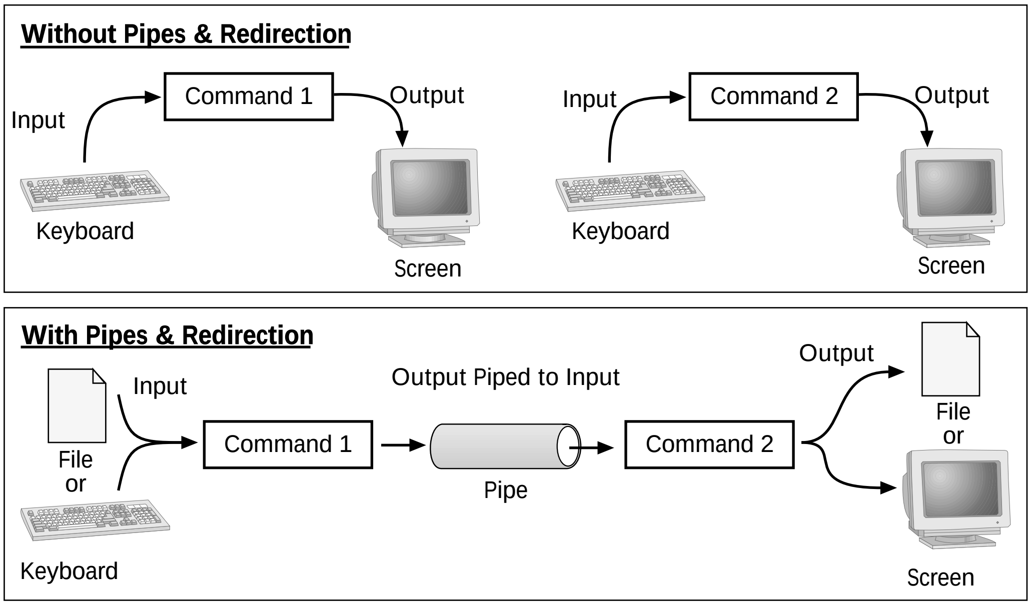


Figure 7-8 Diagram of a pipe between applications

To see a pipe in action, consider two commands: `ls`, which lists the contents of a directory; and `sort`, which sorts all the lines in a file. The `ls` command normally writes output to the screen, whereas `sort` normally requires a filename as a parameter. But you can combine the `ls` and `sort` commands by entering the following at a Linux command prompt:

```
ls -l  | sort
```

The output of this `ls` command is not written to the screen. Instead, it is sent (piped) to the `sort` command. Although the `sort` command normally requires a filename, in this case it receives the names of the files it needs to sort from the `ls` command. The result is that `sort` writes to the screen the lines from `ls`, sorted according to the first word in each line.

By combining the features of STDIN, STDOUT, and STDERR with the ability to redirect these communication channels and use pipes, each Linux utility can interact with other utilities and files to meet the needs of users and system administrators. Table 7-5 shows how input and output can be redirected.

Table 7-5 Redirecting Input and Output

Symbol to use in a command statement	Description	Command-line example
`>` *`filename`*	Writes STDOUT output to the given filename	`ls -l > savelisting`
`>>` *`filename`*	Appends STDOUT output to the given filename (adding it to the end of any existing file contents)	`cat newfile >> existing_data`
`<` *`filename`*	Sends data from the given filename as the STDIN, rather than reading from the keyboard	`my_script < input_codes`
|	Creates a pipe between two programs, so that the STDOUT output from the command on the left of the pipe symbol is used as the STDIN input for the command on the right of the pipe symbol	`ls -l  |  sort`

A savvy system administrator knows command options and useful techniques for combining a few basic commands to provide all sorts of useful information. The next section describes some of these utilities and explains how to combine them.

USING BASIC SYSTEM ADMINISTRATION TOOLS

A good system administrator has a mental "toolbox" of methods for solving problems. A large part of this toolbox is knowing about a number of basic Linux commands that can provide information about the current state of a Linux system and tune the system as needed. This section describes some of these basic utilities. As described in the previous section, these tools are simple in their function, but when combined, they can be powerful and extremely useful.

In later chapters you will learn about many other utilities that are specific to certain tasks, such as backing up files or formatting new hard disks. This section focuses on tools that are useful in a variety of circumstances. If you have worked at a command line in another operating system, some of these tools will be familiar to you. Other tools are unique to Linux and UNIX environments.

Case Sensitivity in Linux

Nearly everything in Linux is case sensitive. This can be a big change for users of other operating systems. It means that typing a command in all capital letters is different from using all lowercase. In fact, each of the items in this list would be a different, distinct command in Linux:

- `find`
- `FIND`
- `Find`
- `FiNd`

Linux commands are all lowercase and must be entered without capital letters.

Filenames and File Extensions

Some operating systems use the last part of a filename as an indication of what type of data the file contains. For example, a file ending in `.gif` is a certain type of graphics file, while a file ending in `.wpd` is a certain type of word processing document. The ending part of a filename is often called the file extension, from the days when a filename consisted of a maximum of eight letters and a three-character extension.

Linux does not use file extensions in this way. Instead, it examines the contents of a file to determine its type. A file may have an extension that indicates its type, but this information should be regarded as a convenience for the user, not a requirement for the operating system. For example, if you create a program using the Perl language, it can be named `myscript.pl`, using the conventional `.pl` file extension, but it can also be named `myscript_written_in_perl` or just `myscript`. None of these filenames will affect whether the script functions correctly.

This leads to another point. Filenames in Linux can be very long—up to 256 characters. They can also contain nearly any character except a forward slash (/). Linux filenames do not have shortened versions; they only exist as the complete filename that you see in a file listing. If you use unusual characters in a filename, such as a space or punctuation marks, you should enclose the filename in quotation marks so that the characters are not interpreted as special commands.

Learning About Linux Commands

In the following sections (and in future chapters), you will learn about many Linux commands. Because most of these commands have numerous options and sometimes complex **syntax** (formatting) rules, you may need to refer to additional information as you work in Linux. The Linux system provides several methods of learning about a command as you work:

- Use the online manual page. These are called man pages in the Linux world. Each **man page** contains a description of the command's syntax and all options supported by the command, along with descriptions of how the command can be used, related commands, and additional information. The man pages rarely contain examples of command usage, however. To view the man page, use the `man` command followed by the command you wish to learn about. For example, to learn about the `ls` command, enter `man ls`.

- Use the `help` option for the command. Most commands will print a summary of options and syntax when you enter the command with `--help` as a parameter. This information is less complete than the man page, but may provide more accessible help, especially if you are already familiar with the command and only need a quick reminder about an option. For example, to learn about the `ls` command, enter `ls --help`.
- Use the **`info`** command. For some commands, the man page is not updated regularly. Instead, an info file is maintained with instructions on the command. For example, to learn about the `ls` command, enter `info ls`.
- Use the **`apropos`** command when you don't know the name of the command to use for a task. The `apropos` command returns a list of man pages that contain a keyword that you enter. For example, to see a list of all man pages that contain information about the LILO boot manager, enter `apropos lilo`. On some systems you must generate a database for the `apropos` command to use before `apropos` will return any helpful results. On a Red Hat Linux system, the command `/usr/sbin/makewhatis` will create such a database.

TIP

You must already know the name of the appropriate command before you can use `man` or the `help` option. If you don't know which command to use to solve a problem, use the `apropos` command or try to find a related word in the index of this book and see which commands that section of the book covers.

Using Regular Expressions

Many times, system administration tasks involve working with patterns of information. These patterns might apply to filenames, information on a Web server, information within database files, or in many other locations and situations. Linux and UNIX use a system of expressing patterns called regular expressions. A **regular expression** provides a flexible way to encode different types of patterns. Regular expressions are used with many of the Linux commands you will learn in this chapter and in later chapters. They are also used in writing shell scripts and other types of scripts and programs that provide system administration functionality. A single regular expression can be used to describe each of the patterns in the following sentences:

- Lines containing the word *President* or *president* (upper- or lowercase *P*)
- Filenames with the digits 18 followed by any other digits
- Text at the beginning of a line that starts with *Cruise* or *cruise* and includes the word *ship* later in the same line
- Filenames that end with TIFF, TIF, Tif, Tiff, tif, or tiff

Regular expressions are similar to using wildcards to define filenames. In DOS or Windows, for example, you can indicate a set of all Word files by using a wildcard statement like this: `*.doc`. Regular expressions can be as simple as `*.doc`, but they can also include more complicated statements. Table 7-6 summarizes parts of a regular expression. It would take an entire book to cover all the nuances of creating regular expressions, but mastering the items listed here will let you take advantage of the power of regular expressions in the Linux commands you will be learning.

Table 7-6 Common Regular Expressions in Linux

Expression syntax	Meaning of syntax
`*`	Match zero or more characters
`?` or `.` (a period)	Match one character
`^`	Match text at the beginning of a line
`$`	Match text at the end of a line
`[abc]`	Match one of the characters in brackets
`[^abc]`	Exclude all of the characters in brackets

To better understand the syntax of regular expressions, study the examples in Table 7-7.

Table 7-7 Sample Regular Expressions

Example regular expression	What it matches
`[iI]tal*`	Any word starting with *I* or *i* followed by *tal*, such as Italy, italy, Italian, italian, Italianate, italianate, and so forth.
`^[mM]us??m`	Text at the beginning of a line that begins with *m* or *M*, followed by *us*, two characters, and *m*. (The two characters are intended to allow misspelled versions of the word *museum* to be found.)
`180[0..9]$`	Text at the end of a line that begins with 180 and ends with a digit from 0 to 9 (any of the years from 1800 to 1809 will match).

Using File and Directory Management Utilities

If you have used other computer systems, the basic file operations needed to access and maintain a system are probably familiar to you. For example, in every operating system, you move and copy files, view the contents of directories, and create new directories. This section describes the utilities that Linux provides for these basic tasks. You can access all of these utilities from any Linux shell (any command-line interface).

Table 7-8 describes each of the basic file system management commands used by Linux. An equivalent command from the DOS/Windows command line is shown for reference. Most of these commands have many options that you can learn about by viewing the man page for the command.

Table 7-8 File System Management Commands in Linux

Command name	Description	Example of command usage	DOS equivalent command
`ls`	List the items in the current directory (or other directory given as a parameter)	`ls`	`DIR`
`cd`	Change the current working directory to the directory given as a parameter	`cd /home`	`CD`
`pwd`	Print working directory (show the current working directory)	`pwd`	`CD` (with no directory name parameter)
`cat`	Dump the contents of a file to the screen	`cat textfile`	`TYPE`
`cp`	Copy files to a new filename and/or location	`cp file.txt file_old.txt`	`COPY`
`rm`	Remove (delete) a file or files given as a parameter	`rm *.txt`	`DEL`
`rmdir`	Remove an empty directory	`rmdir datadir`	`RMDIR` or `RD`
`mv`	Move a file or directory to a new location, effectively renaming it if moved to the same directory	`mv report reportold` `mv report /archive/report`	`REN` or `MOVE`
`mkdir`	Make a new subdirectory	`mkdir archive`	`MKDIR` or `MD`

TIP

Most Linux commands are separate programs that start when you enter their names at a command line. But some commands are built into the shell (the command-line interpreter). These commands do not have a man page—you must read the man page for the shell (enter `man bash`) to learn more about how to use them. The only built-in command from the table of file system commands is `cd`.

Deleting Files in Linux

To delete files in many popular operating systems, you drag or move them to a trashcan or recycle bin. The deleted file remains on your system until you "empty" the trashcan or recycle bin. In effect, when you delete a file in this way, you are not really erasing the file, but simply marking it for deletion when the trashcan is emptied. This serves as a protective system against accidentally deleting files that you discover you need soon after you delete them.

In many cases, even after you have emptied the trashcan or recycle bin, you can use special utilities to "undelete" the erased file, reassembling the contents of the file from your hard disk so that it is a complete file again.

In Linux, files cannot be undeleted except in very rare circumstances. When you use the `rm` command to remove (delete) a file, the file is immediately deleted from your hard disk. Because of the way Linux arranges file information on the hard disk, recovering the pieces of a deleted file is rarely successful. How then can you protect yourself and users on your system from accidentally erasing files that are later needed? Here are a few methods used by some system administrators:

- Always use the `rm` command with the `-i` option, which prompts you for confirmation before deleting a file. This reminds you to reconsider any need for the file you are about to erase.
- Use a **safedelete** utility, which compresses and stores files in a hidden directory when they are "deleted." From this directory, they can be undeleted later if needed. Using this type of utility requires additional maintenance and disk space for the compressed files, but it provides a back-up copy in case deleted files are needed.
- Use the trashcan on one of the popular Linux graphical desktops, such as KDE or Gnome. These act like the trashcan or recycle bin in other popular operating systems: files are not truly deleted until you empty the trash.
- Use a special disk tool that attempts to reassemble the pieces of a deleted file based on their location on your hard disk. Using these tools rarely gives complete success, but can often recover at least part of an important file that was accidentally deleted.

Finding What You Need

Once you know some basic utilities for working with the file system, you can use more complex tools to help you locate information in files and directories. This section describes three such tools: `locate`, `find`, and `grep`.

Use the **`locate`** command to search an index of all files on your Linux system. If Linux finds any directory or filename that matches your `locate` query, it prints the full pathname of that item to the screen. For example, suppose you need to locate the Web server configuration file on a Linux system and can't remember where it is stored. By using the following command, you see a list of all occurrences of `httpd.conf` in the entire Linux file system:

```
locate httpd.conf
```

If you are not certain of the complete name, you can use part of it. If that partial name occurs anywhere in a directory or filename, the `locate` command lists it on screen. For example, you could use the following command to locate the `httpd.conf` file:

```
locate tpd.conf
```

The `locate` command has the advantage of being very fast, because it searches an index of your file system rather than searching the entire file system each time you make a query. Using `locate` has two disadvantages, however. First, if you haven't updated the index since you changed your file system, you may not see the results you need (the item you're looking for may not be listed). Second, if you don't know much about the name you're searching for, the list printed by `locate` can be so large that it's not very useful. You can't use special patterns (regular expressions) to make a more precise query using the `locate` command.

TIP If you leave your Linux system running, the file system index used by `locate` is automatically updated in the middle of the night. Otherwise, you can run the `updatedb` command to update the index (a process that can take several minutes).

The **`find`** command also provides a list of files that match a query string, but it provides many more options than `locate`, so it can be used for much more in-depth and powerful system administration work. The `find` command operates on your file system at the time you run a command; it doesn't use a prebuilt index. This means that other processes might slow down if you run a complex `find` query. The results can also take a few seconds to appear with `find`.

The simplest use of `find` is to search for files that match a specific name pattern and print them to the screen. In this example, the path where the search should begin is given as `/home`, followed by the name of the file to search for and the action to take with each filename found (print it to the screen).

```
find /home -name report.doc -print
```

The `find` command uses full words as options, but preceded by only a single hyphen instead of two. These nonstandard formats can make learning each of the Linux commands a challenge.

The options supported by `find` enable you to perform complex searches for information on your Linux system. For example, using a single (complex) `find` command, you could do any of the following tasks:

- Create an archive file of all the files that have been modified in the last 24 hours
- Delete all files owned by a certain user on the Linux system
- Create a list of all files that are larger than a certain size
- Create a list of all files that have specific access permissions
- Create a list of all files that do not have a valid owner

In later chapters you will see `find` used in examples for specific tasks like those listed here.

The `locate` and `find` commands help you locate a file with specific characteristics. To search within a file, use the **`grep`** command. `Grep` can rapidly scan numerous files for a pattern that you specify, printing out the lines of text that include the pattern. These lines of text can then be processed according to the system administration task at hand. For example, suppose you need to see the shell used by a certain user account. Rather than open a user management tool or look at the /etc/passwd file in a text editor, you can enter this command and immediately see the line of /etc/passwd that contains the information you need:

```
# grep nwells /etc/passwd
nwells:x:564:564::/home/nwells:/bin/csh
```

The last item in the response line indicates that the current default shell for user nwells is the C shell (csh).

TIP The grep command is intended for use with text files, not with binary-format data such as program executables.

You can also perform much more complex searching. Suppose you have a directory full of text files and you want to see all occurrences of a string pattern that starts with ThomasCorp. The following command lists all of those occurrences, showing the filename containing the string and the complete line of text containing the string:

```
grep ThomasCorp  *txt
```

The first parameter—ThomasCorp—is a regular expression. In this case, a specific string is the pattern to search for, with no special characters. When using the grep command, an asterisk is never needed at the beginning or end of the string pattern (such as ThomasCorp*), because grep will locate the string wherever it occurs. For example, grep would find instances of the following strings during the search:

- ThomasCorp
- ThomasCorporation
- ThomasCorps

But these strings would not be included:

- Thomas Corporation
- Thomascorporation
- Thomas Nast

The second parameter to grep is also a regular expression that defines which files to search. The asterisk in the command indicates that all files in the current directory that end with the letters txt should be searched.

The results of the grep command might include lines like these:

- Annual_report.txt: As news of ThomasCorporation reaches customers around the world, we are pleased to…
- memo0518.txt: that Rachel and I think ThomasCorp should be looking seriously at acquiring an interest in…
- meetingsummary.txt: Discussed needs of ThomasCorp to diversify plastics manufacturing capacity for…

The `grep` command is often used with a pipe to search the output from another command. For example, you can pipe the output of the `locate` command through the `grep` command to refine a search. A sample command might look like this:

```
locate tif | grep airframe
```

In cases like this, `grep` uses only a single parameter—the pattern to search for. Rather than include a filename to define the text to be searched, the output of the `locate` command is searched. The results are printed to STDOUT—the screen.

Reviewing System Processes

Linux includes many tools that you can use to track and interact with the many processes that may be running at the same time on your system. Two of these commands are introduced here. These and others are covered in detail in Chapter 10.

The **`ps`** command lists the processes that are currently running on your Linux system. The process list can contain a great deal of information. Selecting various options for the `ps` command lets you control which pieces of information are included in a listing of processes and how that information is organized. The basic format of the `ps` command uses no parameters and produces a listing of programs that you have started in your current session (this is generally a short list, as shown here):

```
$ ps
PID    TTY     TIME        CMD
576    tty1    00:00:00    login
584    tty1    00:00:00    bash
741    tty1    00:00:00    ps
```

In this list, you see a PID (**process ID**) number (a unique number identifying a process); the terminal that the process is using for output (`tty1` is the first main console screen); the CPU time that process has used so far; and the command that started the process.

Other `ps` commands include information such as the user that started (owns) the process, the process priority, current status, and the PID number of the parent process (the process that started this one).

An important command related to processes is the `kill` command. You can use the **`kill`** command to end a process. Chapter 9 explains more about how this occurs within the Linux operating system, but the simplest example of `kill` is shown here, with the PID number of the process you want to end:

```
kill -9 873
```

Chapter Summary

- The role of the Linux system administrator is to keep Linux-based computer systems running efficiently, usually for the use of a group of co-workers.
- A system administrator holds a position of great trust within an organization and must use ethical practices to protect the integrity of the systems being managed.
- Linux systems incorporate many different plain-text configuration files used to set up system services. These files each use a different format. Various graphical tools can be used to configure some services by automatically modifying the appropriate text files.
- Linux administration is built on a collection of single-task utilities that can be combined to achieve the desired results. Understanding these tools and their options is the basis of the toolbox that a system administrator has available to solve problems.

Key Terms

`apropos` — Linux command used to show all man pages that contain a keyword.

chief information officer (CIO) — The executive in an organization who determines how information systems are used within the organization to further its goals or mission effectively.

COAS (Caldera Open Administration System) — A set of graphical utilities developed by Caldera Systems and used to manage many aspects of a Linux system.

daemon — A background process that runs on Linux to handle tasks, such as responding to network traffic, without any visible screen output.

end user — An individual who uses the computer systems in an organization to accomplish assigned tasks, but relies on a system administrator to keep those systems running smoothly.

`find` — Linux command used to search the file system for files matching certain characteristics.

`grep` — Linux command used to search within files for lines containing a certain pattern.

Help Desk — A service in many organizations that assists end users in solving problems related to information technology.

`info` — Linux command used to access online command reference information.

Information Systems Department (IS) — The area of an organization in which the staff are responsible for maintaining computer and information systems that support the employees in their work (also called the IT Department in some organizations).

Information Technology Department (IT) — *See* Information Systems Department.

`kill` — Linux command used to end a process.

LinuxConf — Graphical configuration and administration utility for Linux, developed and supported by Red Hat Software.

`locate` — Linux command used to search an index of the file system for items matching a given pattern.

man page — An online reference documenting a Linux command.

pipe — A connection between two Linux commands (indicated by the | character) that causes the output of one command to be used as the input of a second command.

plain-text configuration file — A file containing human-readable instructions that are used by a program to set its configuration information.

process — A task running on a Linux operating system, managed by the Linux kernel.

process ID (PID) — A number from 1 to 65,000 that is associated uniquely with a process running on a Linux system.

program — An imprecise term used to refer to any process running on a Linux system.

`ps` — Linux command that provides information about processes running on Linux.

redirection — The concept of changing the location where a Linux program receives its input and sends its output in order to increase flexibility and interaction with other Linux programs.

regular expression — A system of expressing patterns using special characters that can be interpreted by many Linux programs.

safedelete — A type of utility that makes files appear to have been deleted but actually saves a compressed copy of each one in case it is needed later.

SAGE (System Administrators Guild) — A professional organization for system administrators.

standard error (STDERR) — The channel used by most Linux programs to send information about errors in program execution.

standard input (STDIN) — The communication channel used by most Linux programs to collect input (normally from the keyboard).

standard output (STDOUT) — The communication channel used by most Linux programs to write output (normally to the screen).

syntax — A formalized arrangement of information to allow a Linux command to understand parameters, options, and so forth.

thread — A piece of a process (or a piece of a daemon, since a daemon is a type of process). The distinction between processes and threads is not important for most system administration work. Instead, the term *process* is used in most cases that don't involve programmers developing software for multiprocessor computers.

utility — An imprecise term referring to a program used to administer a computer system rather than do work for an end user.

YAST — A graphical configuration utility developed by the makers of SuSE Linux.

REVIEW QUESTIONS

1. Why are nontechnical skills like curiosity and creativity important to being a successful system administrator?
2. Name one key advantage of using multiple plain-text configuration files in Linux.
   a. They are compatible with configuration files from other operating systems.
   b. Several system administrators can access the same configuration file at the same time.

   c. If one configuration file becomes corrupted, none of the other system services are affected.
   d. Special utilities are required to change system configuration settings.

3. Which of these tasks is not likely to be assigned to you as a system administrator?
   a. Develop a new cash register system using C programming.
   b. Install new hard disks in Linux servers.
   c. Teach new users how to access their e-mail accounts.
   d. Attend a conference on improving system security.
4. A ________________ runs a subtask as part of a larger task, often on a multi-processor system.
   a. daemon
   b. process
   c. thread
   d. utility
5. Name two disadvantages to using any of the current graphical Linux configuration/administration tools.
6. Name three graphical tools that are used either for general system administration or for administration of a specific service (such as a Web server).
7. Which of the following does not have correctly formed options?
   a. `ls --help`
   b. `ls --color -R`
   c. `ls -il -aX --reverse`
   d. `ls -sort=time`
8. A pipe is a method of connecting processes with daemons. True or False?
9. The command `ls | sort` causes the following to occur:
   a. The output of the `ls` command is sent to the `sort` command. The results are printed to the screen.
   b. It cannot be determined without information about the next command to be executed.
   c. The output of the `ls` command is written to a file named `sort`.
   d. The output of the `ls` command is filtered based on the regular expression contained in the file `sort`.
10. A regular expression is used to:
    a. define a list of threads that a process can execute
    b. assign values to variables
    c. define a complex pattern used for searching
    d. build filenames from component parts

11. When you run a program called `gather_data`, it normally reads lines entered at the keyboard. If you use the command `gather_data < input_text` to run the program, the following occurs:
    a. The `gather_data` command is executed followed by the `input_text` command.
    b. The input that the `gather_data` program would normally read from the keyboard is taken from the `input_text` file instead.
    c. The `input_text` program runs first, collecting data, which is then passed through a pipe to the `gather_data` program.
    d. Both `gather_data` and `input_text` run as concurrent processes reading from the keyboard as STDIN.
12. Describe the difference between `find` and `locate`.
13. The regular expression `[cC]hapter0[12345]*` will *not* match which of the following files:
    a. `chapter01`
    b. `Chapter03.doc`
    c. `Chapter1.doc`
    d. `Chapter02`
14. The `find` command should be used instead of the `locate` command when:
    a. your locate index has not been updated recently
    b. the number of processes on the system is large
    c. you prefer to use `grep` at the same time
    d. the file system appears to be unstable
15. The `grep` command is *not* useful for which of the following:
    a. Searching for all filenames that match a pattern
    b. Determining which directories are currently in use
    c. Finding lines of text that contain a certain word
    d. Locating specific information in the output of another command
16. The `locate` command uses a prebuilt index of your file system to search for file information. True or False?
17. Describe the difference between `ps` and `kill`.
18. Linux filenames can be eight characters with a three-character extension, but a longer filename is also stored for reference. True or False?

19. The `rm` command is used in Linux to:
    a. remove special characters from a filename
    b. delete files from a hard disk
    c. remove case-sensitivity settings
    d. manage regular expressions
20. Only command names are case sensitive in Linux. True or False?
21. Name three methods of learning about Linux commands as you work at a command line.
22. All Linux systems use a trashcan facility to save deleted files. True or False?
23. Describe the errors in this command: `Ls /help`
24. The `ps` command does *not* provide information about:
    a. who started a process
    b. the number assigned to a process
    c. the current status of the process
    d. when the threads of the process expire
25. The `kill` command is used to end:
    a. a process
    b. a thread
    c. a user account
    d. a locate query

HANDS-ON PROJECTS

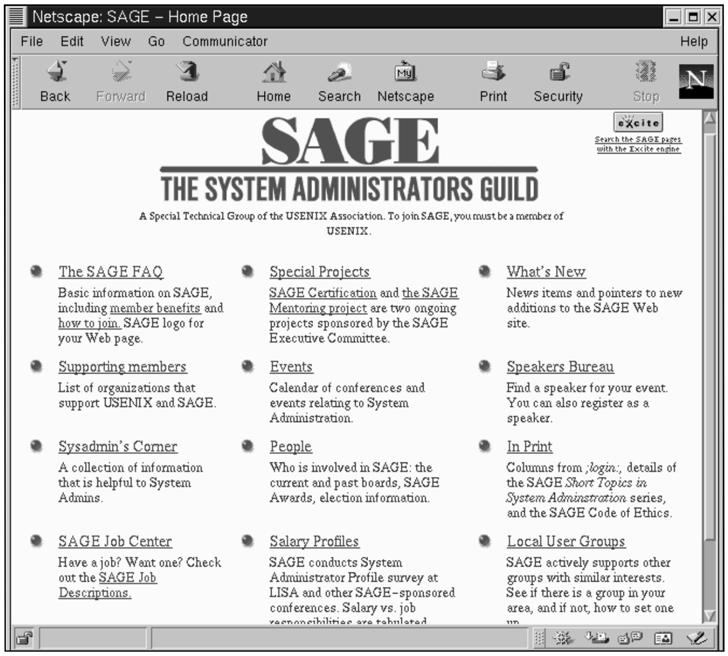
Hands-On Project

Project 7-1

In this activity you access the SAGE Web page to learn about conference events for system administrators. As you learn about the events, you can see what topics are presented to system administrators seeking to improve their skills. To start the exercise, you should be at a computer with an Internet connection and a Web browser.

To find information about system administrator conferences:

1. Enter **http://www.usenix.org/sage/** on the Location or Address line of your browser. The SAGE Web page appears, as shown in Figure 7-9. The SAGE Web page contains information for professional system administrators.
2. Click the **Events** link on the SAGE Web page. A list of recent and forthcoming events is displayed. (The events listed change over time.)
3. Click the link for one of the listed events near your area. (You may also choose to select a conference event with a topic that is of special interest to you, such as security or the Web.)
4. Locate a link on the page for the event that you have chosen that describes the program for the conference. The name of this link varies with each event. You may also choose a link that shows information about past conferences and their topics.


Netscape: SAGE – Home Page
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
excite
Search the SAGE pages with the Excite engine

SAGE
THE SYSTEM ADMINISTRATORS GUILD

A Special Technical Group of the USENIX Association. To join SAGE, you must be a member of USENIX.

- The SAGE FAQ
  Basic information on SAGE, including member benefits and how to join. SAGE logo for your Web page.
- Special Projects
  SAGE Certification and the SAGE Mentoring project are two ongoing projects sponsored by the SAGE Executive Committee.
- What's New
  News items and pointers to new additions to the SAGE Web site.
- Supporting members
  List of organizations that support USENIX and SAGE.
- Events
  Calendar of conferences and events relating to System Administration.
- Speakers Bureau
  Find a speaker for your event. You can also register as a speaker.
- Sysadmin's Corner
  A collection of information that is helpful to System Admins.
- People
  Who is involved in SAGE: the current and past boards, SAGE Awards, election information.
- In Print
  Columns from *;login:*, details of the SAGE *Short Topics in System Administration* series, and the SAGE Code of Ethics.
- SAGE Job Center
  Have a job? Want one? Check out the SAGE Job Descriptions.
- Salary Profiles
  SAGE conducts System Administrator Profile survey at LISA and other SAGE-sponsored conferences. Salary vs. job responsibilities are tabulated
- Local User Groups
  SAGE actively supports other groups with similar interests. See if there is a group in your area, and if not, how to set one

Figure 7-9 SAGE Web page

Project 7-2

In this activity you practice using some of the commands described in this chapter. To complete this project, you should be logged on to a standard Linux system using a regular user account.

To practice Linux commands:

1. Type **`cp /etc/termcap ~/testfile`** to create a practice file. This copies a system file to your home directory.
2. Type **`ls -l`** to list the files in your home directory. Make certain the test file is listed. Note the file's size and the date it was created.
3. Type **`mkdir archive`** to create a subdirectory to hold the file.
4. Type **`mv testfile archive`** to move the sample file to the new directory.
5. Type **`cd archive`** to change your working directory to the new directory.
6. Type **`pwd`** to check which directory you are working in.
7. Type **`ls -l`** to list the contents of the new directory.
8. Type **`grep Linux testfile`** to search the test file for the string `Linux`.
9. Type **`grep Linux testfile > results`** to repeat the search, but this time write the results to a file.
10. Type **`cat results`** to review the contents of the results file.
11. Type **`rm -i testfile results`** to delete the test file and results file. What does the `-i` parameter do? How would you now delete the archive directory?

Project 7-3

In this project you view information about the processes running on your Linux system. To complete this project, you should be logged on to a standard Linux system using a regular user account.

To view information about Linux processes:

1. Type **`ps`** to list the processes that you have started in your current Linux session.
2. Type **`ps ax`** to list all of the processes running on your Linux system for all users.
3. Type **`ps ax | more`** to view the `ps` output one screen at a time. Note that this command uses a pipe to combine the `ps` command with the `more` command.
4. Type **`ps ax | grep httpd`** to search the output of the `ps` command. This command uses `grep` to find out the number of Web servers (`httpd` daemons) running on your Linux system.

Case Projects

1. You've been working at Tyson Electronics as a system administrator for about three months. Most of the employees are highly trained in technical topics and use their computers for all of their daily work, which includes sending e-mail to colleagues around the world.

   One of the product managers approaches you with a belligerent tone and insists that you improve the system response time so he can download large e-mail messages faster. You realize that increasing the speed of the company's Internet connection involves a substantial cost. How could you respond to the product manager's request while maintaining a good relationship with all employees? Describe the probable effects on your relationships with all employees and the company's success if you were to retaliate against the belligerent manager by slowing down his connection or creating other technical problems on his account. What if your actions were discovered?

2. The CIO of Big Brother Corp. is concerned that some employees are spreading rumors about the company's financial status to colleagues in other companies. She asks you to collect all the e-mail messages sent by two employees who are under the most suspicion so they can be reviewed by management. The messages are archived and available to you as system administrator; you also have the ability to capture new messages as they are sent. The company has a policy stating that e-mail is subject to review, but no one really expects that others will read their mail. What is your reaction to the demand of the CIO? Do you feel you have an ethical obligation to remind employees of the corporate policy so they are more careful in their use of company resources? If the company didn't have a policy about reading employees' e-mail and you left the company because of an incident such as this, what would you tell your next potential employer about why you left?

3. While checking the available free hard disk space on the server, you notice an employee using an inordinate amount of disk space. On examining a few of the employee's numerous files, you discover that the majority contain offensive material. What action might you take towards this employee? How will the company's stated policies regarding employee privacy and use of company resources affect your actions? Are your own actions subject to review?